Міністерство освіти і науки України

Національний технічний університет України

«Київський політехнічний інститут ім. Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Системне програмування

Лабораторна робота №7

«Програмування операцій ділення чисел»

Виконав:

студент групи ІО-82

Шендріков Є. О.

Залікова № 8227

Перевірив Порєв В. М.

Київ – 2020

Мета

Навчитися програмувати на асемблері ділення чисел, вивчити перетворення з двойкової у десяткову систему числення.

Завдання

1. Створити у середовищі MS Visual Studio проект з ім'ям **Lab7**.

2. Написати вихідний текст програми згідно варіанту завдання. У проекті мають бути три модуля на асемблері:

- головний модуль: файл **main7.asm**. Цей модуль створити та написати заново, частково використавши текст модуля main5.asm попередньої роботи №5;

- другий модуль: модуль **module** попередньої роботи №6;

- третій модуль: модуль **longop** попередньої роботи №6.

3. Додати у модулі процедури, які потрібні для виконання завдання. Обгрунтувати розподіл процедур по модулям.

4. У цьому проекті кожний модуль може окремо компілюватися.

5. Скомпілювати вихідний текст і отримати виконуємий файл програми.

6. Перевірити роботу програми. Налагодити програму.

7. Отримати результати – кодовані значення чисел згідно варіанту завдання.

8. Проаналізувати та прокоментувати результати, вихідний текст та дизасембльований машинний код програми.

Варіант завдання

1. Потрібно запрограмувати на асемблері вивід значення факторіалу $n!$ У **десятковому коді**. Використати програмний код обчислення факторіалу попередньої лабораторної роботи №5. Для кожного студента своє значення n

$$\boldsymbol{n = 30 + 2 \times H},$$ де H – це номер студента у журналі.

2. Перетворення у десятковий код запрограмувати діленням числа підвищеної розрядності на 10. Ділення на 10 виконати двома способами – діленням "у стовпчик" та діленням груп бітів.

3. Програмний код ділення "у стовпчик" та ділення групами бітів оформити у вигляді процедур. Процедури повинні містити такі параметри: адреса ділимого, розрядність ділимого, значення дільника (B) та інші (за необхідності).

4. Запрограмувати на асемблері також обчислення формули, яку вибрати з таблиці, наведеної нижче.

Мій варіант

25	$y = \frac{x}{11} 2^{-m}$

n = 30 + 2 * 25 = 80

Текст програми

main7.asm

```
.586
.model flat, stdcall
include \masm32\include\kernel32.inc
include \masm32\include\user32.inc
include module.inc
include longop.inc
includelib \masm32\lib\kernel32.lib
includelib \masm32\lib\user32.lib

option casemap :none

.data
	Caption db "80!" ,0
	Caption1 db "Значення функції за варіантом" ,0

	textBuf dd 100 dup(?)
	textBuf1 dd 60 dup(?)

	var dd 25 dup(0)
	x dd 80
	y dd 0

	test1 dd 25 dup(4294967295)
	test1res dd 50 dup(0)
```

```
	test2 dd 25 dup(4294967295)

	test31 dd 25 dup(4294967295)
	test32 dd 25 dup(0)
	test3res dd 50 dup(0)


.code
	main:
	;Факторіал в десятковій формі
	mov [var], 1
	@fact:
		push offset var
		push x
		call Mul_Nx32_LONGOP
	dec x
	jne @fact

	push offset textBuf
	push offset var
	push 400
	call Str_Dec
	invoke MessageBoxA, 0, ADDR textBuf, ADDR Caption, 0

	;Обчислення функції
	push -1320880352    ; X
	push 4              ; m
	call Func

	mov y, eax

	push offset textBuf1
	push offset y
	push 32
	call Str_Dec

	invoke MessageBoxA, 0, ADDR textBuf1, ADDR Caption1, 0
	invoke ExitProcess,0
end main
```

longop.asm

```
.586
.model flat, c

.data
	x dd 1
	n dd 0

	num10 db 10
	inner dd 0
	num7 db 7
	minn db 0
	spacee db 3


.code

;алгоритм множення N*32 з попередньої лабораторії
Mul_Nx32_LONGOP proc
	push ebp
	mov ebp, esp
```

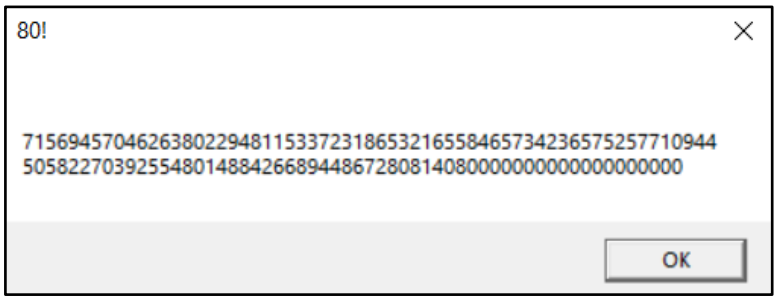
```
	mov edi, [ebp + 12]
	mov ebx, [ebp + 8]
	mov x, ebx
	mov n, 25

	xor ebx, ebx
	xor ecx, ecx
	@mult32:
		mov eax, dword ptr[edi + ecx]
		mul x
		mov dword ptr[edi + ecx], eax
		clc
		adc dword ptr[edi + ecx], ebx
		mov ebx, edx

		add ecx, 4
		dec n

		jnz @mult32

	pop ebp
	ret 8
Mul_Nx32_LONGOP endp

;ділення у стовпчик
Div_Column_LONGOP proc
	xor ebx, ebx
	xor ecx, ecx

	dec edx
	cmp byte ptr[esi + edx], 0
	jnz @cycleout
	inc bl

	@cycleout:
	mov ch, byte ptr[esi + edx]
	@cycleinner:
	shl cl, 1
	shl bh, 1
	shl ch, 1
	jnc @zero
	inc bh
	@zero:
	cmp bh, num10
	jc @less
	inc cl
	sub bh, num10
	@less:
	inc inner
	cmp inner, 8
	jnz @cycleinner
	mov byte ptr[esi + edx], cl
	mov inner, 0
	sub edx, 1
	jnc @cycleout
	ret
Div_Column_LONGOP endp

;вивід в десятковій системі
Str_Dec proc
	;процедура StrHex_MY записує текст шістнадцятькового коду
	 ;перший параметр - адреса буфера результату (рядка символів)
	;другий параметр - адреса числа
```

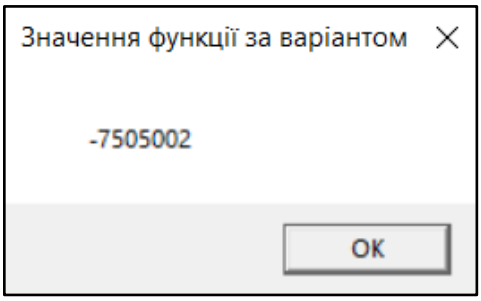
```
	;третій параметр - розрядність числа у бітах (має бути кратна 8)
	push ebp
	mov ebp,esp
	mov edx, [ebp+8]		;кількість бітів числа
	shr edx, 3			;кількість байтів числа
	mov esi, [ebp+12]		;адреса числа
	mov edi, [ebp+16]		;адреса буфера результату

	mov eax, edx
	shl eax, 2

	mov cl, byte ptr[esi + edx - 1]
	and cl, 128
	cmp cl, 128
	jnz @plus
	mov minn, 1
	push edx
	@minus:
	not byte ptr[esi + edx - 1]
	sub edx, 1
	jnz @minus
	inc byte ptr[esi + edx]
	pop edx
	@plus:

	@cycle:
	push edx
	call Div_Column_LONGOP
	pop edx
	add bh, 48
	mov byte ptr[edi + eax], bh
	dec eax
	cmp bl, 0
	jz @cycle
	dec edx
	jnz @cycle

	cmp minn, 1
	jc @nomin
	mov byte ptr[edi + eax + 1], 45
	dec eax
	@nomin:

	inc eax
	@space:
	mov byte ptr[edi + eax], 32
	sub eax, 1
	jnc @space

	pop ebp
	ret 12
Str_Dec endp

end
```

module.asm

```
.586
.model flat, c
.data
num11 dd 11

.code
    ;процедура StrHex_MY записує текст шістнадцятькового коду
    ;перший параметр - адреса буфера результату (рядка символів)
       ;другий параметр - адреса числа
       ;третій параметр - розрядність числа у бітах
Func proc
       push ebp
       mov ebp,esp
       mov ecx, [ebp+8]           ;m
       mov eax, [ebp+12]          ;X

       xor edx, edx
       mov ebx, eax
       shl ebx, 1
       jnc @plus
       sub edx, 1
       @plus:

       idiv num11
       sar eax, cl

       pop ebp
       ret 8
Func endp

End
```

Результати роботи програми





Аналіз результатів

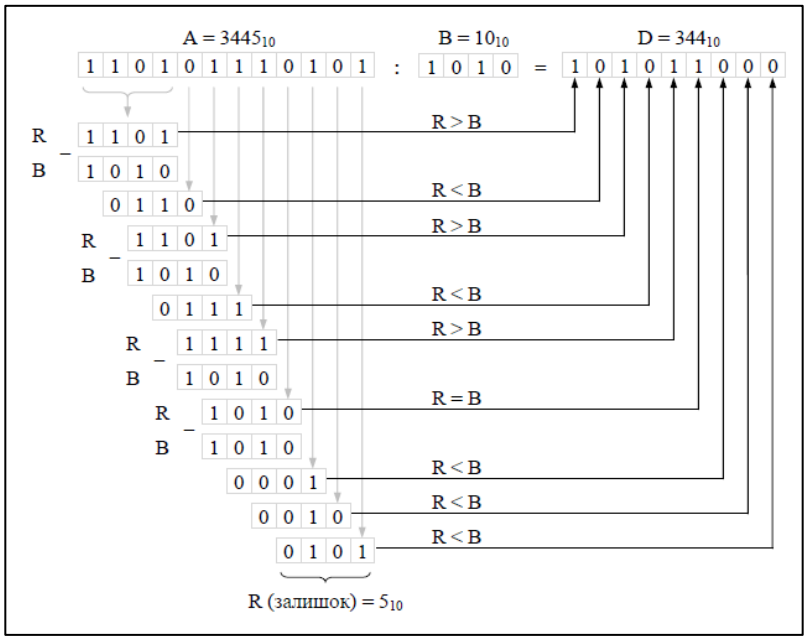
80! = 7156945704626380229481153372318653216558465734236575257710944
505822703925548014884266894486728081408000000000000000000000

$$y = \frac{x}{11} * 2^{-m} = \frac{-1320880352}{11} * 2^{-4} = -7505002$$

Результати програми повністю збігаються з теоретичними даними. Програма працює вірно.

Обчислення заданої функції здійснюється за допомогою комбінації операцій IDIV, SHL та ADD. Зсув вліво використовується для множення на степінь двійки. Ділення вбудованою командою IDIV застосовується для ділення зі збереженням знаку. Для команди IDIV явно вказується тільки один операнд – дільник. Виконання команди IDIV визначається розрядністю цього операнду. Якщо результат ділення від'ємний, то часткове та залишок записуються як від'ємні числа у додатковому коді відповідної розрядності. Дана команда використовує алгоритм ділення групами бітів.

Ділення у стовпчик відбувається за таким алгоритмом:



1. Встановити $i = n - 4$.
2. Взяти чотири старші біти числа A. Позначити це як $R = \{a_{n-1}, a_{n-2}, a_{n-3}, a_{n-4}\}$
3. Якщо R більше, або дорівнює B, то: і-та цифра (біт) результату d_i дорівнює 1, віднімаємо $R = R - B$ інакше: i-та цифра результату d_i дорівнює 0
4. Зменшити i на одиницю. Якщо i менше нуля, то кінець роботи.
5. Помножити R на два. Таке множення означає зсув бітів на одну позицію вліво. Додати до R у молодший розряд i-й біт числа A, тобто a_i.
6. Перехід на п. 2.

Ділення групами бітів ґрунтується на тому, що багаторозрядне ділиме у позиційній системі числення – у тому числі й у двійковому коді, може бути записане у вигляді суми груп розрядів.

Розглянемо алгоритм ділення 16-бітового числа A на 8-бітове число B. 16-бітове число A можна представити у вигляді суми двох 8-бітових груп A_1 та A_0. Якщо потрібно, щоб результат ділення (D) був також 16-бітовим (наприклад, щоб запобігти переповнення розрядної сітки при B=1), то додамо зліва ще одну фіктивну 8-бітову групу A_2 з усіх нулів – число A від цього не зміниться:

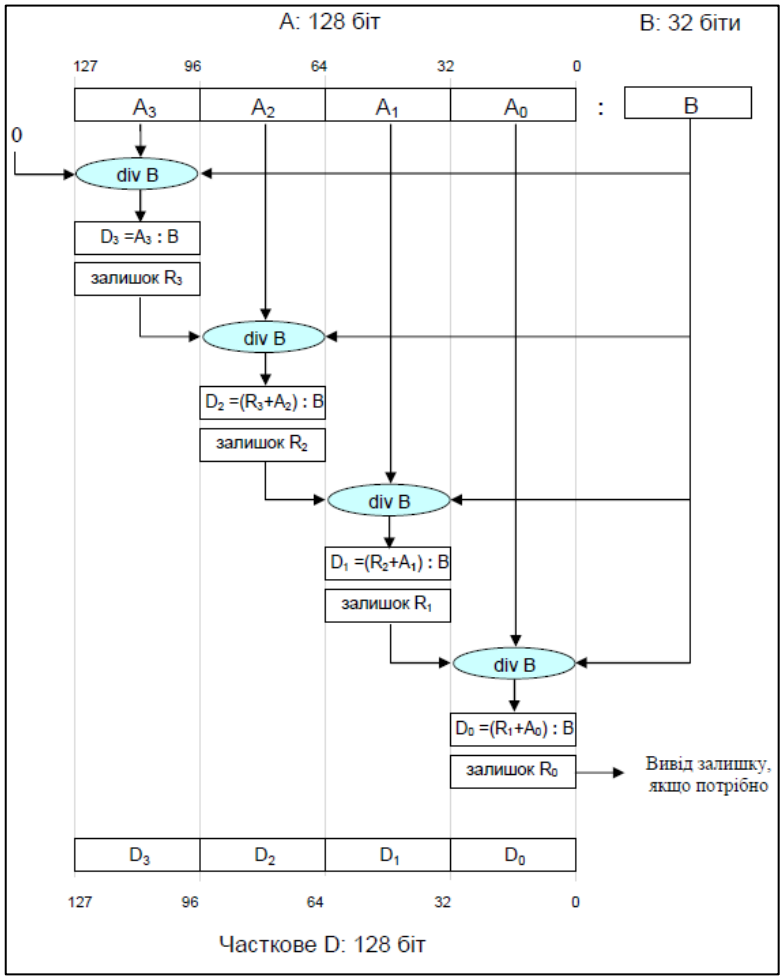
A	=	A_1	A_0	=	$A_2 = 0$	A_1	A_0
16 бітів		8 бітів	8 бітів		8 бітів	8 бітів	8 бітів

Математично операцію ділення D = A/B можна записати у наступному вигляді:

$$D = \frac{2^{16}A_2 + 2^8A_1 + A_0}{B} = 2^8 \frac{2^8A_2 + A_1}{B} + \frac{A_0}{B}$$

Можна уявити собі пристрій, який ділить 16-бітове число на 8-бітове число. $D_1 + \frac{R_1}{B}$

Результат цілочисельного ділення цілих чисел без знаку буде у вигляді двох 8-бітових цілих чисел: часткового D та залишку R.



На даному рисунку показаний приклад ділення групами бітів 128 розрядного числа на 32 розрядне число.

Висновок

Під час виконання лабораторної роботи я закріпив навички програмування на Асемблері, а саме: створення процедур, підключення модулів, механізм циклів. Також я реалізував операції множення, ділення та переводу у десяткову систему числення для чисел підвищеною розрядністю. Навчився програмувати побітові операції, вивчив основні команди обробки бітів. Кінцева мета роботи досягнута.